A Research Agenda to Improve the Quality & Performance of Commoditized, Open-source Software

Adam Porter
aporter@cs.umd.edu
University of Maryland

Emerging Opportunities and Challenges

- Emerging Opportunities
  - *Commoditization* of infrastructure software
    - Decreased time-to-market at lower cost to consumer
    - High demand for s/w, but low per-unit profit for suppliers
    - Technology convergence & standardization
      - Increasingly difficult to justify unique s/w solutions
  - Maturation of open-source *development processes & products*
    - Web browsers/servers – e.g., Mozilla, Apache
    - Operating systems– Linux, FreeBSD
    - System/Network support tools – e.g., Sendmail, Bind, Samba
    - Middleware – e.g., ACE+TAO, OmniOrb
  - Emergence of open-source *virtual communities*
    - Where many globally distributed—but Internetworked—community members contribute resources, knowledge, & time

- Persistent Challenges
  - Quality of functionality
  - Quality of service
  - Usability
  - Cost
- New Challenges
  - High code volatility
  - High platform heterogeneity
  - Enormous configuration & optimization space
  - Razor-thin margins

What can we do to make open-source software better, faster, ***and*** cheaper forever?

Solution Approach: Distributed Continuous Testing & Profiling

- Leverage open-source virtual communities to incrementally & opportunistically improve quality & performance by engaging users in continuous testing and profiling
  - Regression testing & profiling widely distributed & conducted in parallel on machines provided by open source community during off-peak hours
    - Syntactic correctness – clean compile
    - Semantic correctness – regression testing
    - Performance measurements – memory footprint, throughput, latency
  - User community resources are coordinated carefully – *i.e.,* follow the sun around the world
  - Adapt testing & profiling based on results of earlier testing & profiling
    - Precisely identify broken configurations
    - Automate error detection via rollback
- Key constraints
  - Minimize human developer effort
  - Minimize end-user overhead
  - Avoid compromising privacy & security

Candidate Research Agenda

Research Challenge	Solution Approach	Recommendation
▪ Software research historically limited by suitability & availability of artifacts	▪ Leverage open-source virtual community resources to focus on "real world" software ▪ *e.g.*, artifacts, test cases, test case outputs, CMS logs	▪ Sponsor open-source software projects as a research enabler ▪ Encourage sponsored projects to instrument
▪ The enormous platform/feature configuration space greatly complicates software testing & optimization efforts	▪ Leverage open-source virtual community resources to improve quality & performance of software systems ▪ *i.e.*, previous slide	▪ Sponsor research on network-centric open-source development ▪ Distributed testing ▪ Distributed profiling ▪ Network-enabled development tools